Palestine Polytechnic University

College of Information Technology and Computer Engineering

Interactive Game Using Kinect Sensor
(Hero Kinect)

Team members:

Shahd Madhoun Madeha Tahboub

Supervisor name:

Mohammad Jabari

2021-2022

إهداء

بسم الله الرحمن الرحيم

(وَقُل رَّبِّ زِدْنِي عِلْمًا)

صدق الله العظيم

إلى معلم البشرية سيدنا محمد عليه افضل الصلاة والسلام.

إلى صاحب السيرة العطرة، (والدي الحبيب)، أطال الله في عُمره.

إلى من وضعتني على طريق الحياة، لما لها من الفضل ما يبلغ عنان السماء (أمي الغالية)، أدامها الله.

إلى إخوتي؛ من كان لهم بالغ الأثر في كثير من العقبات والصعاب.

إلى أصدقائي الذين طالما كان المر يزول معهم.

إلى جميع أساتذتي الكرام؛ ممن لم يتوانوا في مد يد العون لي.

إلى وطني الحبيب فلسطين.

إليهم جميعاً أهدي هذا الجهد المتواضع.

Abstract

In recent years, technology has taken over our lives. Everything we do is reliant on technology, particularly in the aftermath of the corona outbreak last year. As a result, technology has a huge impact on all members of society, particularly children. So, this game attempts to increase children's movement while playing video games.

One of the most important characteristics of the game is that it relies on the child's movement, maintains a safe distance between the player and the screen so that his vision is not harmed and exposed to issues such as obesity, isolation, and others.

The game revolves around a character who must jump, move left and right to avoid obstacles. At the same time the player must collect coins. What is special about this game is that it allows children to be the controller and feel as if they are actually in the game.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Overview

Our project is an interactive game aimed for children. The game encourages children to move and stand away from the screen, allowing them to enjoy their time more safely.

The game depicts barriers and obstacles for the player to overcome by jumping and moving in front of the sensor and screen, then the sensor will read the player's movement and display it on the screen so that the player's character is visible on it.

1.2 Motivation and Importance

Nowadays It's fairly common to find a child under five years old playing with one of these phones; some parents even use this magical device to distract their children, despite the fact that it has a negative impact on their health.[1]

In addition, the corona epidemic altered people's lifestyles and introduced certain new habits. One of the effects of the pandemic was that children began to spend more time on their smartphones than they had previously, resulting in less movement. And it's a problem that needs to be addressed.[2]

So we want to create an interactive game that is free of the negative impacts that a traditional game has. By developing a game that can be played without being too near to the screen, requires movement and at the same time the player doesn’t feel he is different and away from technology.

1.3 Scope of the Project

The project is an interactive game that targets female and male children who are at least one meter tall and over five years old.[3] Also the target device for the game is mainly Xbox one devices and laptops with windows operating system and Kinect SDK downloaded on it.

1.4 Objectives

1. Develop an interactive game.
2. The game displays on the screen virtual obstacles that the player needs to avoid.
3. Game players can gain a score by collecting the coins that appear on the screen.
4. Keeping players eyesight safe while playing by maintaining a safe distance between them and the screen.
5. While playing, the player's body should move, the game can detect the player's physical movements.

1.5 Timeline/ Project Scheduling

We will follow the Software Development Life Cycle to achieve the objectives. And as shown in table 1.1 these are the tasks that we have to do. Also table 1.2 shows the number of weeks required to achieve each task.

Table 1.1: Project Tasks.

Task number	Task name	The time required in weeks
1	System definition and planning.	5
2	Determine the Project requirements.	4
3	Description of the project Requirements.	4
4	System design.	4
5	System development and programming.	6
6	Integration and system testing.	3
7	Documenting the website.	Along the working period

Table 1.2: Gantt Chart.

	First semester				Second semester		
Weeks	2-6	7-10	11-14		2-6	7-11	12-14
System definition and planning							
Determine the Project Requirements							
Description of the project Requirements							
System design							
System development and programing							
Integration and system testing							
System documentation							

Table Key
Estimate time to finish the task
Real time to finish the task
Holiday between semesters

1.6 Project Description

Our idea intends to create an interactive video game that encourages children to move while playing a video game. The Kinect sensor will be used to track the players' body movement and collect data. Then the data will be used to display a character inside the game that reflects the player's movement in real life. Also, we will use Unity engine and C# programming language to build the game, control player movement, and represent sensor data.

1.7 Expected Result

An interactive game that detects physical movements of the player.

1.8 Overview of the Document

In this chapter we reviewed the problems that children face these days. Also, we gave an overview of the project and a short description. In next chapters we will explain requirements specification and software design.

Chapter 2: Background

2.1 Overview

This chapter briefly describes the theoretical background of the project and relative projects. Also, a short description of the Kinect sensor.

2.2 Theoretical Background

After the appearance of video games game reviews began appearing, as well as articles examining the market for video games. Books and researchers started asking what games were and why people played them. And since then a lot of researchers have been studying games.
In The Video Game Theory Reader book the writer lists elements that are the heart of what makes the video game a unique medium, and need to be addressed in any discussion of them. The most fundamental of these elements are: an algorithm, player activity, interface, and graphics.
So the interface and graphic of the game should be attractive to the player since it's what the player sees mostly. Also player activity is important, which cares about the movement of the player in real life and the character movement in the game. Lastly, the algorithm is also categorized into four basic elements: responses, representation, rules and randomness. All of the four categories are important but randomness is the most important since it keeps the game interesting and the user does not expect what comes next.[4]

2.3 Literature Review

Beat Saber game.

Description:

Beat Saber is a virtual reality rhythm game. You have a blue and a red lightsaber in each hand. When the music starts, blocks come across you, and you must hit.[5]

Features:

It aims to make the player move more while playing. So there is more movement than usual games.

Limitations:

Wearing virtual reality glasses requires the gamer to remain close to the screen. The glasses are also quite weighty. In addition, the player wearing them will be unaware of what is going on around him in real life.

2.4 Kinect Sensor

Kinect is a line of motion sensing input devices produced by Microsoft and first released in 2010. Kinect sensors contain RGB cameras, infrared projectors and detectors that map depth through either structured light or time of flight calculations, which can in turn be used to perform real-time body skeletal detection. They also contain microphones that can be used for speech recognition and voice control.[6]

The camera in Kinect is powered by both hardware and software. It also does two things:

- generates a three-dimensional moving image of the objects in its field of view.
- recognizes moving humans among those objects.

Figure 2.1 shows how the Kinect sensor works in steps.

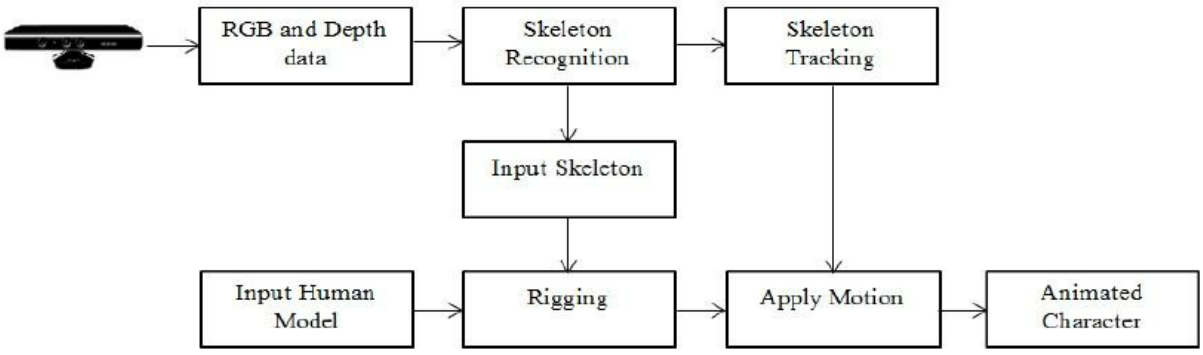


Figure 2.1: How Kinect Works.[7]

Once the sensor captures the information, they are immediately processed by artificial intelligence software. It is able to classify the different objects that are within a scene. It recognizes humans by its head and limbs.[8]

Chapter 3: Requirements Specification

3.1 Functional requirements

The functional requirements were identified in the early stages of development of the project, and include the following:

3.1.1 Functional requirements as user requirement

- The player shall be able to start the game.
- The player shall be able to exit the game.
- The player shall be able to view the help interface.
- The player shall be able to customize the difficulty level.
- The player shall be able to view a bar that indicates the player's health during the game.
- The player shall be able to view the highest score in the game.
- The player shall be able to play again after the game is finished.

3.1.2 Functional requirements as system requirement

- The game will display an avatar that will mimic the user's actions.
- The avatar will interact with game features to earn points.
- The game will end when a player's strength reaches 0% or the road ends.
- The game will be able to track only one person in the workspace.

3.2 Non-functional requirements

Regarding non-functional requirements, our work focused on the following:

3.2.1 Performance Requirements

- The system will be able to catch your motion between 4 ft (7 in / 1.4 m) in front of the Kinect sensor for one player.[9]
- The system should run in real-time.
- The system will respond to the player's motion in at most 70 ms.[10]

3.2.2 Software Quality Attributes

- **Ease of Use:** Someone with little to none technical experience in the operations of electronics should be able to set up and use this system by following a simple set of instructions.
- **Ease of Learning:** The learning curve for this software should be short since the software should perform the corresponding tasks based on natural human motions. The player should be able to use all the system's functions after reading instructions at the help interface.
- **Responsiveness:** The system's efficiency is evident in the speed with which it warns the player when he hits the barrier, as well as the reduction of the player's health at each collision, in order to increase the game's efficiency and effectiveness. So the system will respond in at most 70 ms.

3.3 Domain requirements

The game is playable on Xbox one or devices equipped with a Kinect sensor. While playing there is a need to assure that the Kinect is properly plugged in.

3.4 Context Diagram

Our game has two main sides which are the player and the sensor. The player interacts with the sensor depending on what he sees on the screen (game interface). After that the sensor will read the player's motion and transfer it to our system. Then the game will show a character based on the data.
Figure 3.1 shows the components of our system and the interaction between them.

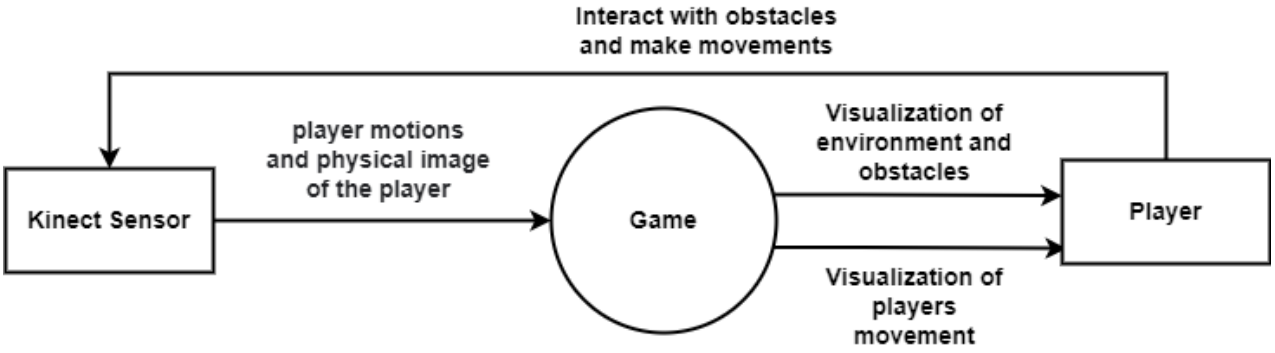


Figure 3.1: Context Diagram.

3.5 Use Case Diagram

The interactions between the system and its actors, in this case the player, are depicted in Figure 3.2. So, it demonstrates how the player interacts and uses the system.

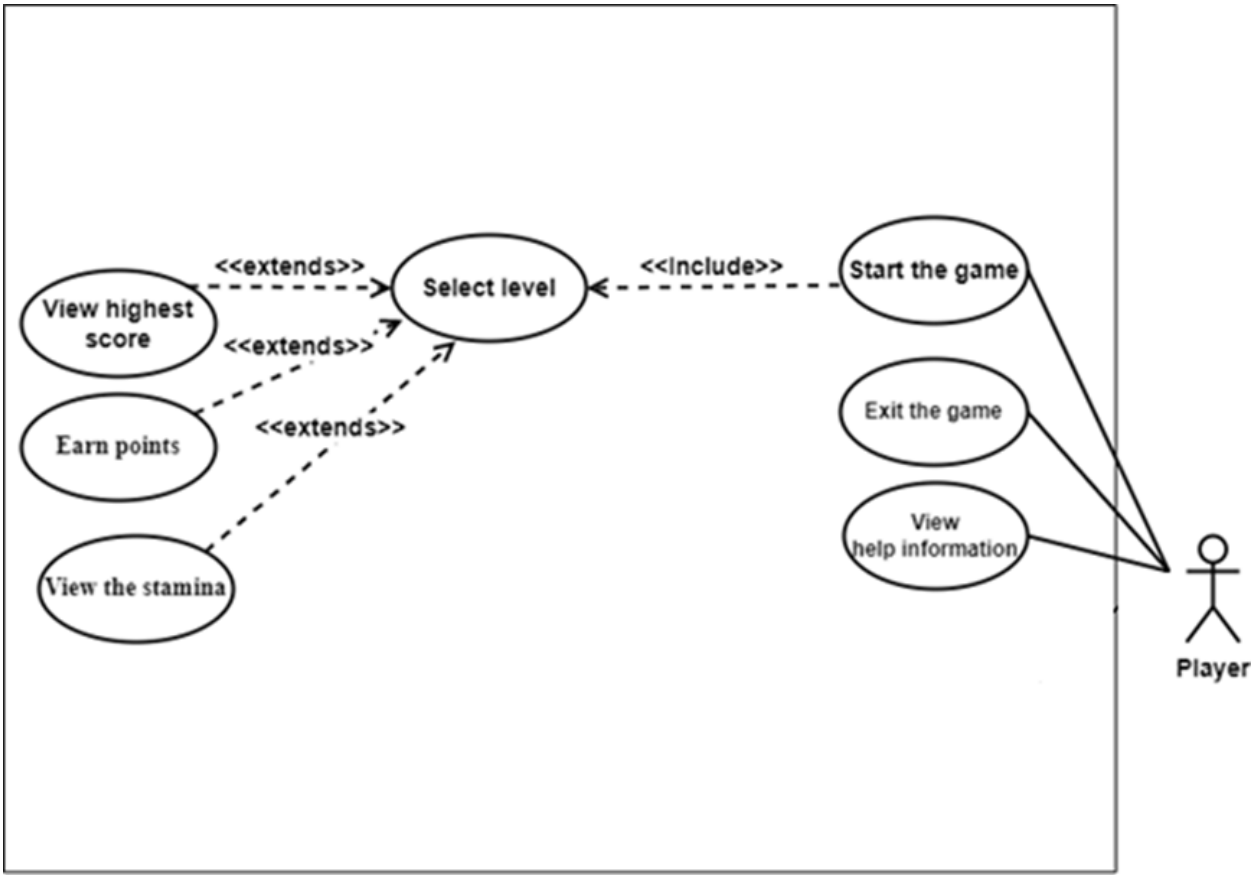


Figure 3.2: Use Case Diagram.

3.6 Functional Requirement Analysis

Table 3.1 shows the details of starting the game functionality.

Table 3.1: Start Game.

Use Case	**Start the game.**
Goal	Go to the select level menu to start the game.
Preconditions	To be at the main menu screen.
Scenario	1) The player chooses the start button by specific movement with his hand. 2) A screen with levels will appear to choose between them.
Exceptions	The sensor can't read player's movement

Table 3.2 shows the details of exiting the game functionality.

Table 3.2: Exit the Game.

Use Case	**Exit the game**
Goal	Ending the game
Preconditions	The user should be at the main menu screen.
Scenario	1) The player chooses the exit button by specific movement with his hand. 2) The game will exit.
Exceptions	The sensor can't read player's movement

Table 3.3 shows the details of viewing help information functionality.

Table 3.3: View Help Information.

Use Case	View help information
Goal	To ensure that the player understands all of the game's instructions.
Preconditions	The user should be at the main menu screen.
Scenario	1) The player chooses the help button by specific movement with his hand 2) Help information will appear on the screen.
Exceptions	The sensor can’t read player’s movement

Table 3.4 shows the details of selecting the level functionality.

Table 3.4: Select the Level.

Use Case	Select level
Goal	The player selects the level difficulty.
Preconditions	The user should be at the level's menu screen.
Scenario	1) Three levels will be displayed as options. 2) The player selects one by specific movement with his hand.
Exceptions	The sensor can’t read player’s movement

Table 3.5 shows the details of earning points functionality.

Table 3.5: Earn Points.

Use Case	Earn points.
Goal	To add excitement to the game.
Preconditions	Start playing the game.
Scenario	1) Coins will appear on the screen while playing. 2) The player can earn them by moving his body or his hands and touching the coins in the game.
Exceptions	The sensor can't read player's movement

Table 3.6 shows the details of viewing the highest score functionality.

Table 3.6: View Heights Score.

Use Case	View highest score
Goal	To add excitement to the game.
Preconditions	Begin playing and earn coins as the player goes.
Scenario	The highest score will be displayed while playing the game.
Exceptions	The sensor can't read player's movement

Table 3.7 shows the details of viewing the health functionality.

Table 3.7: View the Health.

Use Case	**View the health**
Goal	To add excitement to the game.
Preconditions	Start playing the game
Scenario	1) The player will be able to view his health while playing. 2) The player must avoid the barriers that appear in front of him; however, if he does not cross the barriers and collides with them, his health will be reduced by 4% for each collision.
Exceptions	The sensor can’t read player’s movement

Chapter 4: Software Design

4.1 Introduction

This chapter includes the following: game layout, class diagram, game architecture, sequence diagram as well as the graphical interfaces. With an explanation for each.

4.2 Game Layout

The game's components should be organized or laid out in a specific order. First, the position of the sensor should be between 0.6 m and 1.8 m from the floor, the higher the better. The sensor should be placed within 15 cm above your television (screen). In addition, the player should take a step back roughly 1.4 meters. In addition, the sensor should be connected to either an Xbox One or a laptop, with the latter being connected to the screen. Figure 4.1 shows the layout of the game.

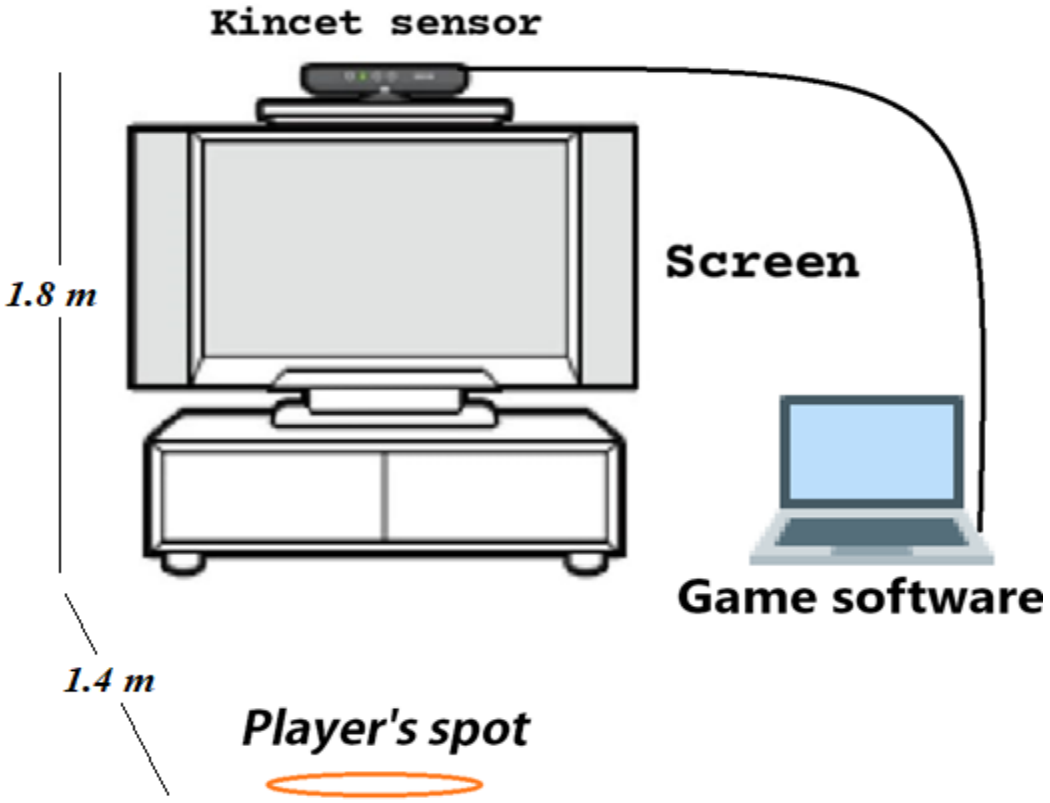
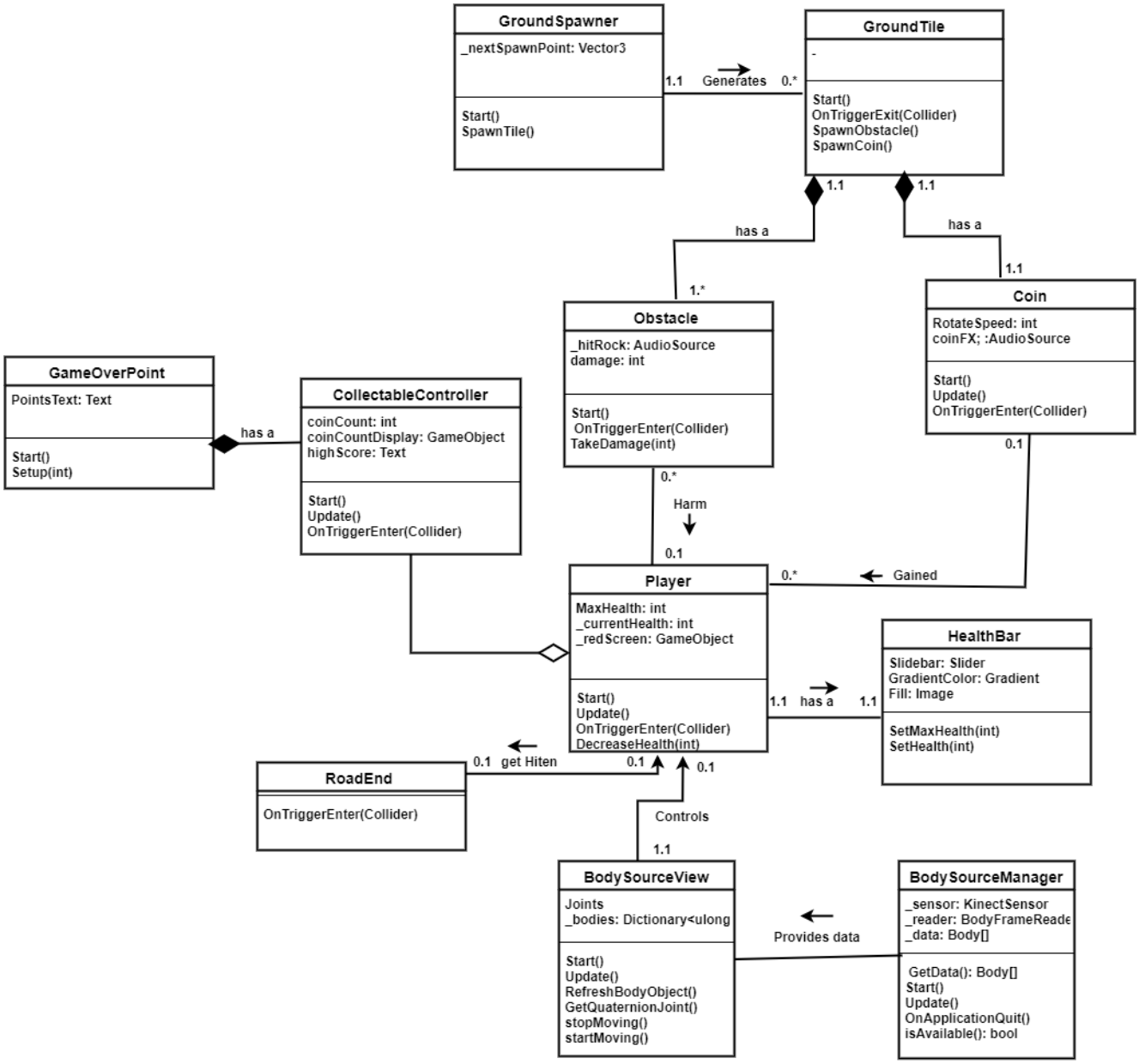


Figure 4.1: Game Layout.

4.3 Class Diagram

Figure 4.2 shows the main classes in our project:



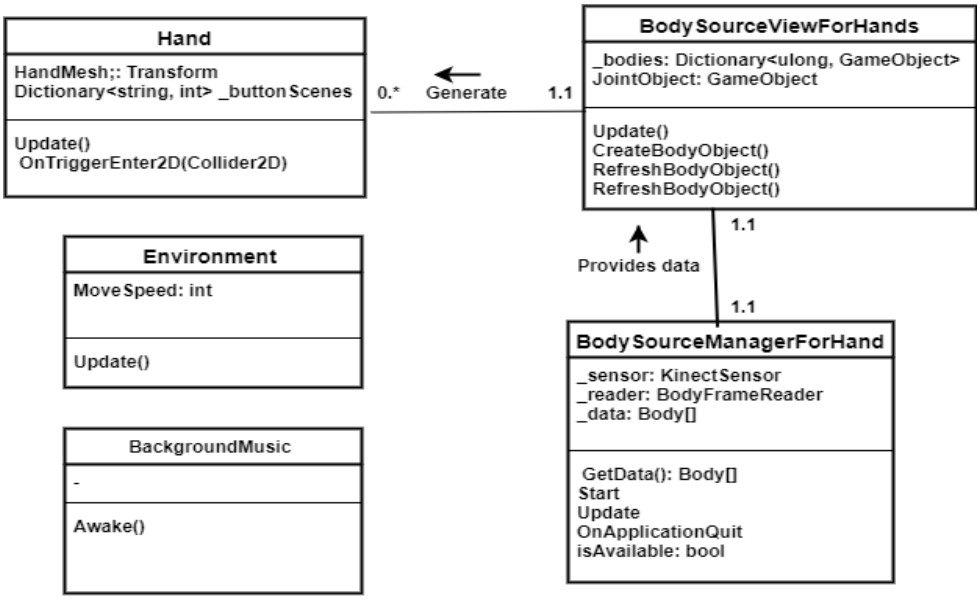


Figure 4.2: Class Diagram.

4.4 Sequence Diagrams

The diagram in Figure 4.3 shows the interaction between the user, game and the sensor. While the sensor is on the sensor will detect the player's movement and bodySourceManager will get the data from the sensor. And bodySourceView will create a game object from the data that bodySourceManager got.

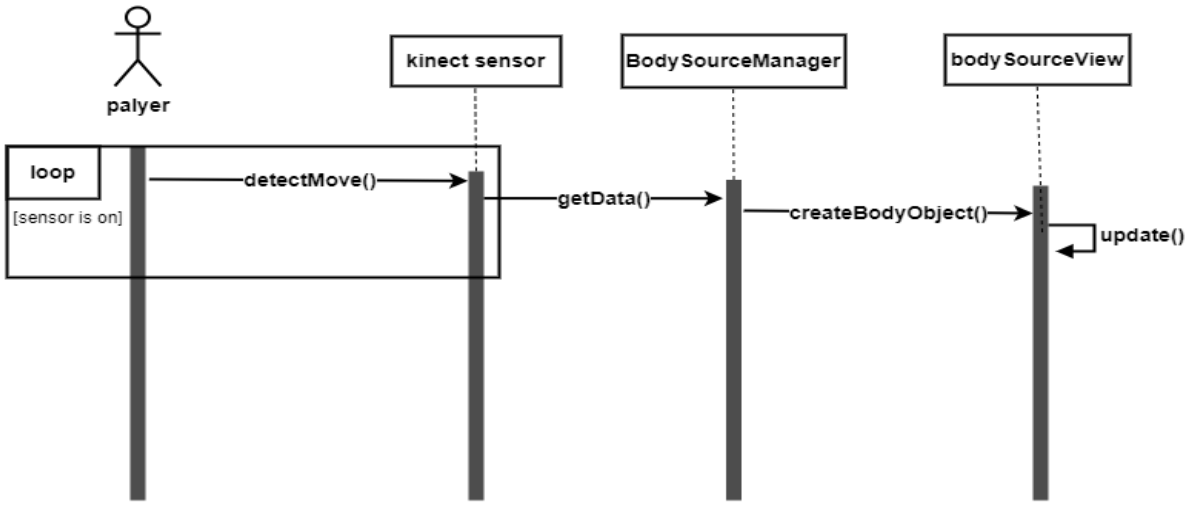


Figure 4.3: Sequence Diagram.

4.5 Storage

For this game the data needed to be stored is just the highest score. So instead of using a relational database we will use a file based management system to store the data. The data will be stored as key value (dictionary).

The file is used because the data we need to store has no relation and even in future we can add as many keys and values as we want.

4.6 User Interface

4.6.1 Theoretical

The game that we want to build is directed for children so the interface that we need to build should be simple and attractive. So we have to choose attractive colors and a simple environment scene. Also the character that we need to choose should look kind.

Humans are trichromats, which means they can see three primary colors: blue, green, and red. The retina in the human eye can detect light with wavelengths ranging from 400 to 700 nanometers, which is known as the visible spectrum.

Each primary color corresponds to a different wavelength, beginning with blue (400 nanometers) and ending with red (700 nanometers).

Green is located in the middle of the spectrum, at approximately 555 nanometers. This wavelength is optimal for our perception. Because of its location in the spectrum's center. As a result, we chose green as the primary color in the game.

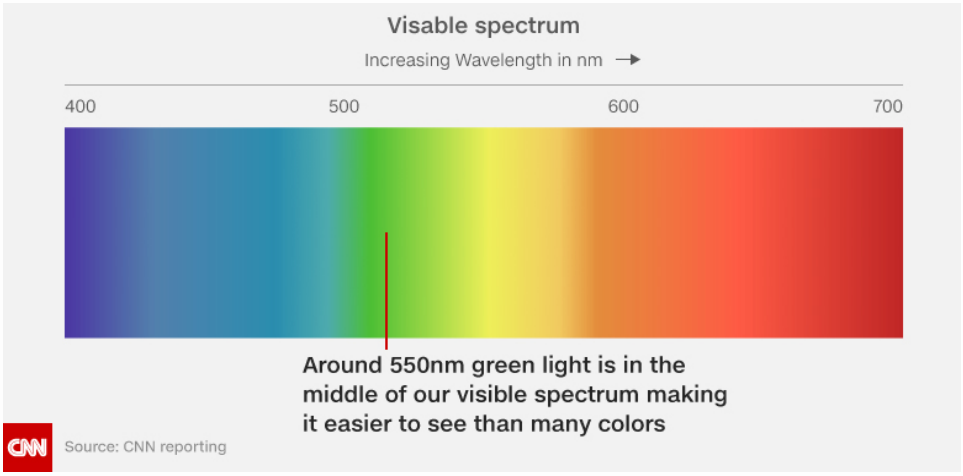


Figure 4.4: Colors Of Light.[11]

4.6.2 Page Design:

We created a basic interface for the game using Adobe XD based on the functional requirements that we decide:

1. **Home Page**

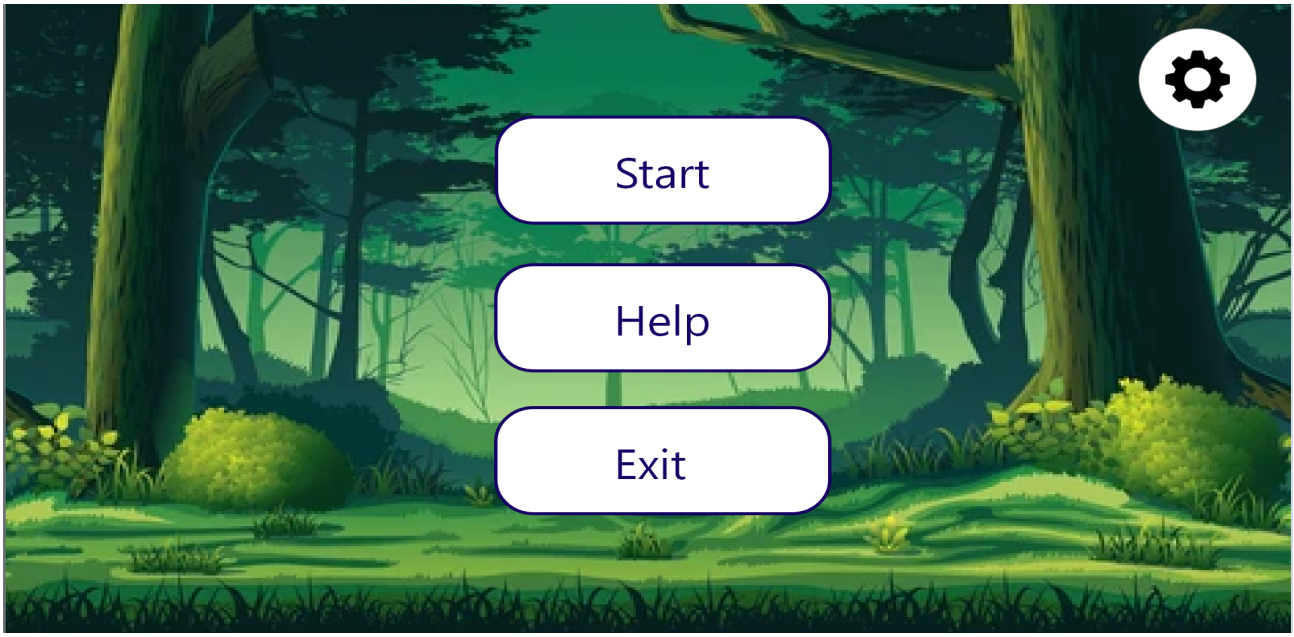


Figure 4.5: Home Page.

2. Help Page

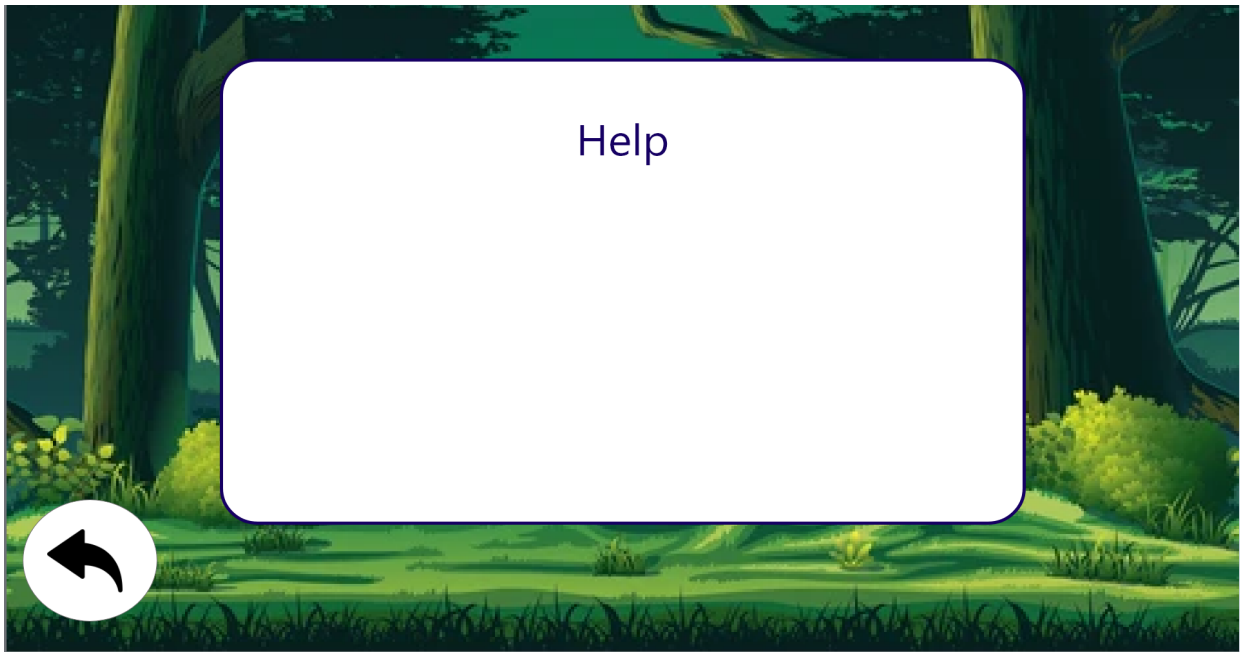


Figure 4.6: Help Page.

3. Choose Difficulty Page

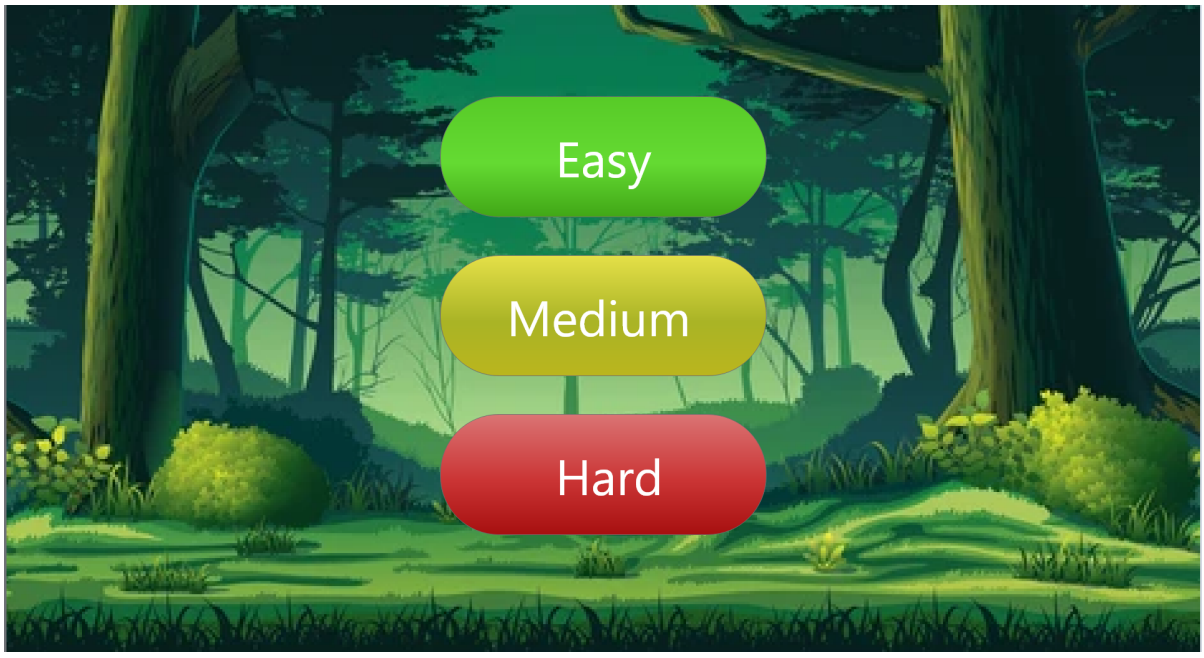


Figure 4.7: Choose Difficulty Page.

4. Game Page

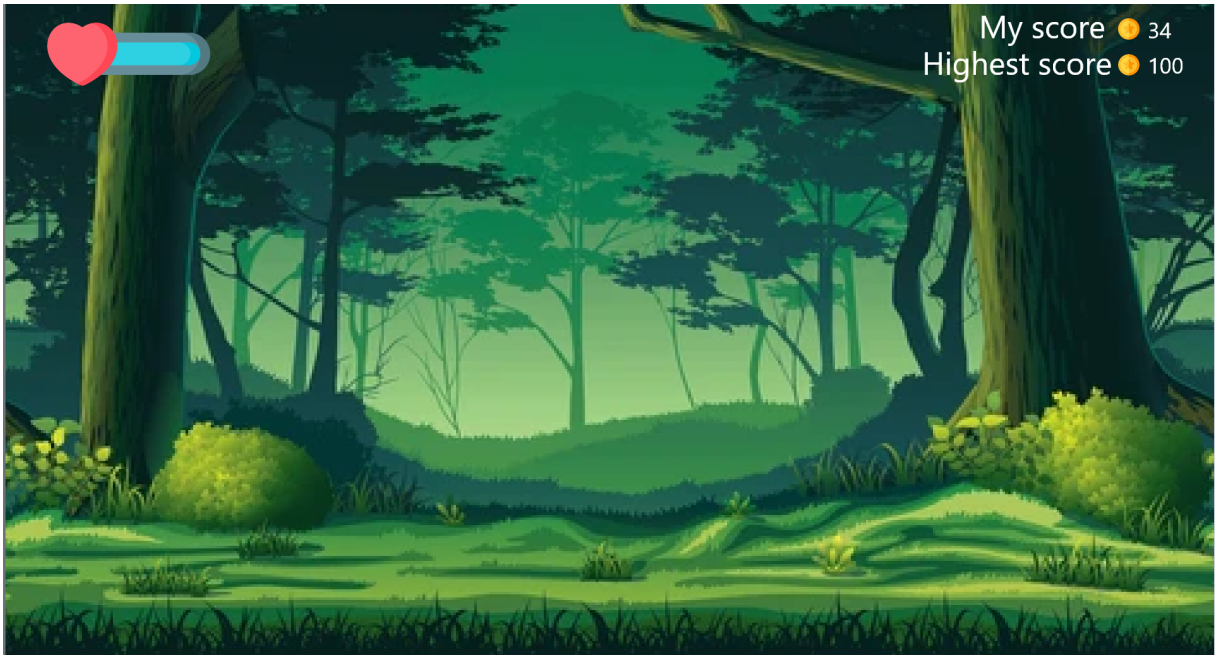


Figure 4.8: Game Page.

5. **Game Over Page**

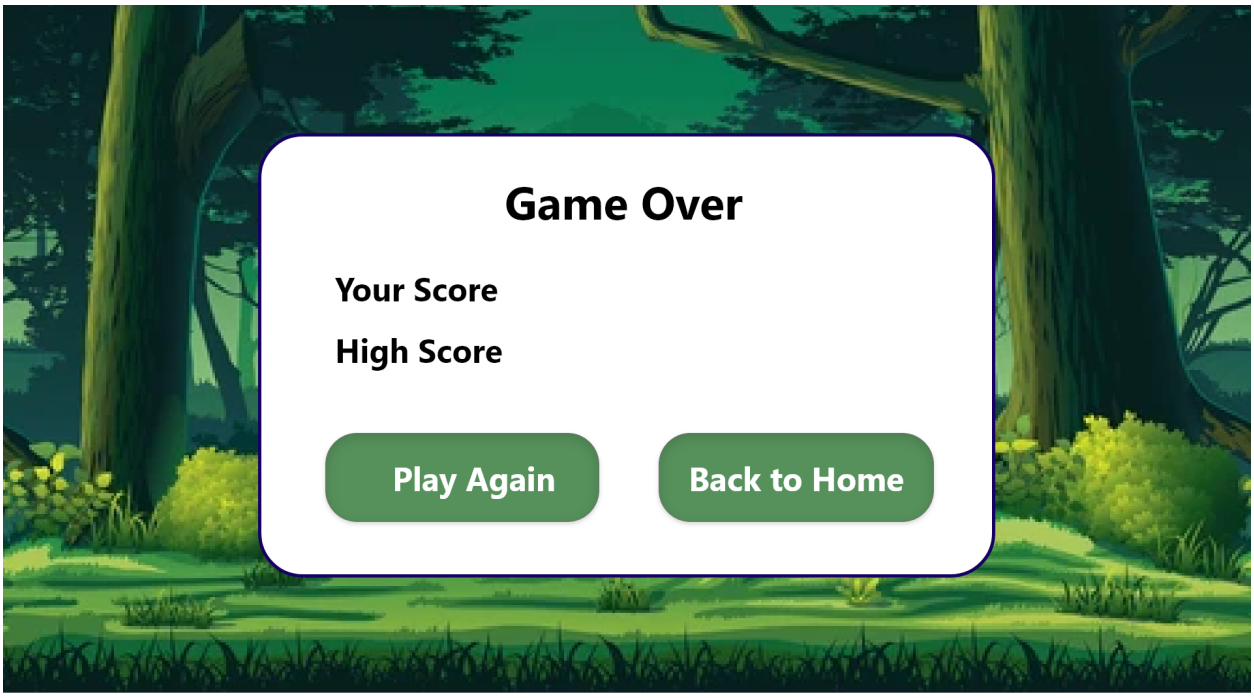


Figure 4.9: Game Over Page.

Chapter 5: Software Demonstration:

5.1 Introduction

This chapter will discuss the technologies that were used in building the game. Also some of the implementation details and issues that faced the project during the implementation.

5.2 Technology And Tools Used

1. **Kinect for Windows SDK 2.0**

   The Kinect for Windows Software Development Kit (SDK) 2.0 enables developers to create applications that support gesture and voice recognition, using Kinect sensor technology on computers running Windows 8, Windows 8.1, and Windows Embedded Standard 8.[12]

2. **Unity Engine**

   Unity is a 3D/2D game engine and powerful cross-platform IDE for developers.

   As a game engine, Unity is able to provide many of the most important built-in features that make a game work. That means things like physics, 3D rendering, and collision detection. From a developer's perspective, this means that there is no need to reinvent the wheel. Rather than starting a new project by creating a new physics engine from scratch–calculating every last movement of each material, or the way light should bounce off of different surfaces.[13]

   Also one of the most important reasons for choosing Unity is because it supports using Kinect so we downloaded Unity Pro packages which are available on Microsoft's official website and support building Kinect based Unity apps. So, we used some of its classes after changing a little to suit our functionality.[14]

3. **Unity Assets Store**

   Asset Store: A Unity website that lets you download premade assets for use in your game development projects. Unity developers can also publish their own premade assets to the Asset Store for other developers to download and use.[15]

   Unity Asset Store was used in the project to download the tree, obstacles and skateboard prefabs.

4. **C# Programming language**

C# is a highly expressive, object-oriented programming language used for a broad range of multi-platform software. Developers often write in C# to build projects intended for long-term use, simply because it's easy to maintain its code. Apart from its universality, the strength of C# lies in the fact that it is a modern C language, making it easy to update, upgrade, and rebuild your software's backend.[16] Also C# is the language used in Unity.

5. **Visual Studio**

It is a source code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.[17]

6. **GitHub**

GitHub is a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration.[18] Also It offers the distributed version control and source code management functionality of Git.

GitHub and Git were as important as other technologies that we used. Each update in the project was pushed on github with a message that represents the changes made. So we can go back for any version of the project we want.

7. **Mixamo**

Mixamo is a free online service for automatically rigging and animating 3D characters. In the website there are characters for every situation. Each character is fully textured and rigged, ready to use in a creative project right away. Characters and animations can be downloaded in a variety of formats for use in motion graphics, video games, film, or illustration. Export improvements will keep the project lean and efficient.[19] So Mixamo was used to download the character used in the game.

5.3 Implementation Details

During implementation we always tried to follow the conventions used in Unity engine and C# language. In the Unity engine we made a folder structure and divided files into categories. And for the code we tried to write clean and optimized code. Also C# naming conventions were followed. Implementation can be divided into three main parts as follow:

5.3.1 Character And Motion

After downloading Unity Pro packages to build Kinect-based Unity apps we imported it to our application. Two main classes were used. First class is body source manager which is used to get the data form the Kinect sensor and get the state of it (connected or not). The second class that we used from Unity Pro packages is body view manager which is responsible for connecting the data coming from the sensor with the player or character joints. And for that we studied the name of the joints in human bodies and the degree that each of them rotate and created them on the player after animation rigging.

Figure 5.1 shows the character after animation rigging has been applied. The character's bones are represented by the blue color. The red squares represent joints of the characters and the target for the two bone constraints. Finally green squares represent the hints for the two bone constraints which control how the middle node bends.

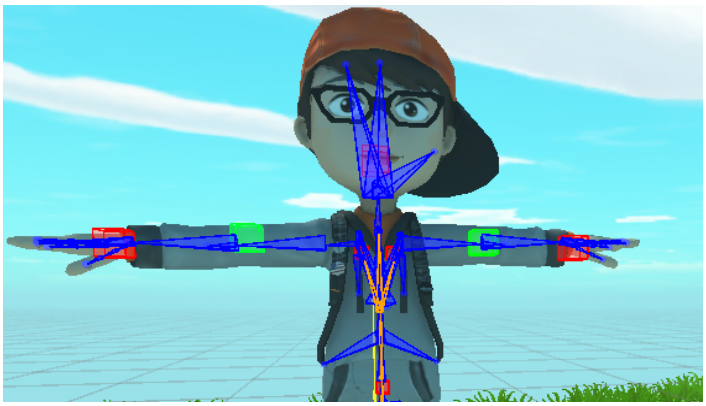


Figure 5.1: Character Joints And Bones.

5.3.2 Prefabs

After making the game scene environment we had to add the coins and obstacles in the environment. So instead of adding them statically we wrote a code to dynamically add the coins and obstacles in random positions and choose random obstacle types.

As a result, we created Prefabs for the obstacles and coins because we use them multiple times in the scenes. This is preferable to simply copying and pasting the GameObject because the Prefab system allows us to automatically keep all the copies in sync and any edits we make to a Prefab Asset are automatically reflected in the instances of that Prefab[20].

5.3.3 Coins and Obstacles

When the character hits coins his score will increase and when he hits obstacles his health will decrease and a scene to show the damage will be shown. Also each time the character passes the coins or obstacles they will disappear from the environment to make the game faster. Also, when the player hits an obstacle or a coin a sound will be played.

Figure 5.2 shows that the properties for coins and obstacles and the most important property is ‘Is Trigger’, This feature allows us to determine whether the player has hit the obstacle or the coin so that we can perform the appropriate special operations, as they are linked to the OnTiggerEnter function, this function in which we carry out the operations required for each case.

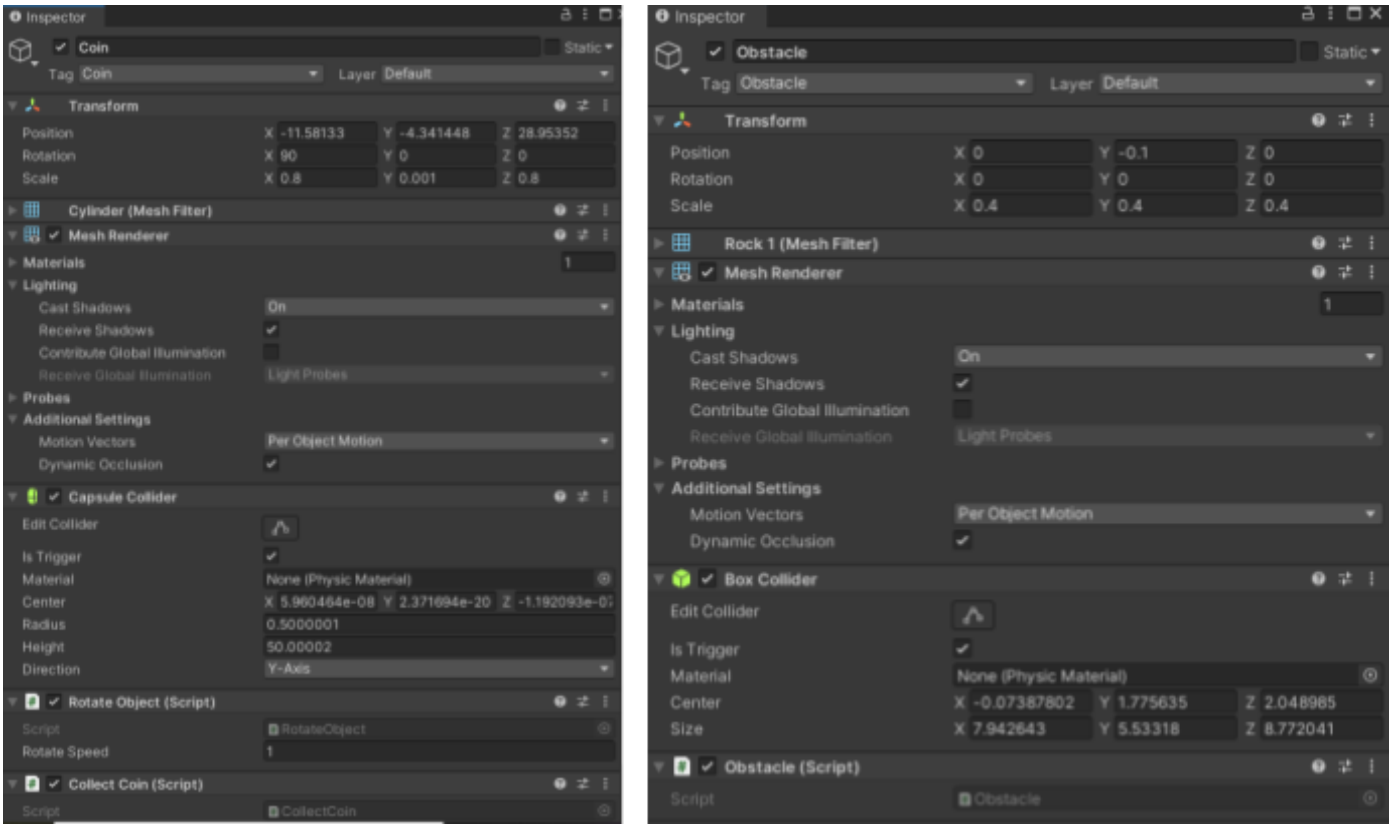


Figure 5.2: Coins & Obstacle Properties.

5.3.4 Health Bar

When the game begins, the color of the bar will be green, indicating that the player has high health. When the player hits an obstacle, his health decreases and the bar fills up with gradient color. Furthermore, the color changes from green to yellow, orange, and then red depending on the player's health. The bar's color is red when the health is minimal to indicate danger.

Figure 5.3 shows the properties for health bars such as width, height and gradient color..etc.

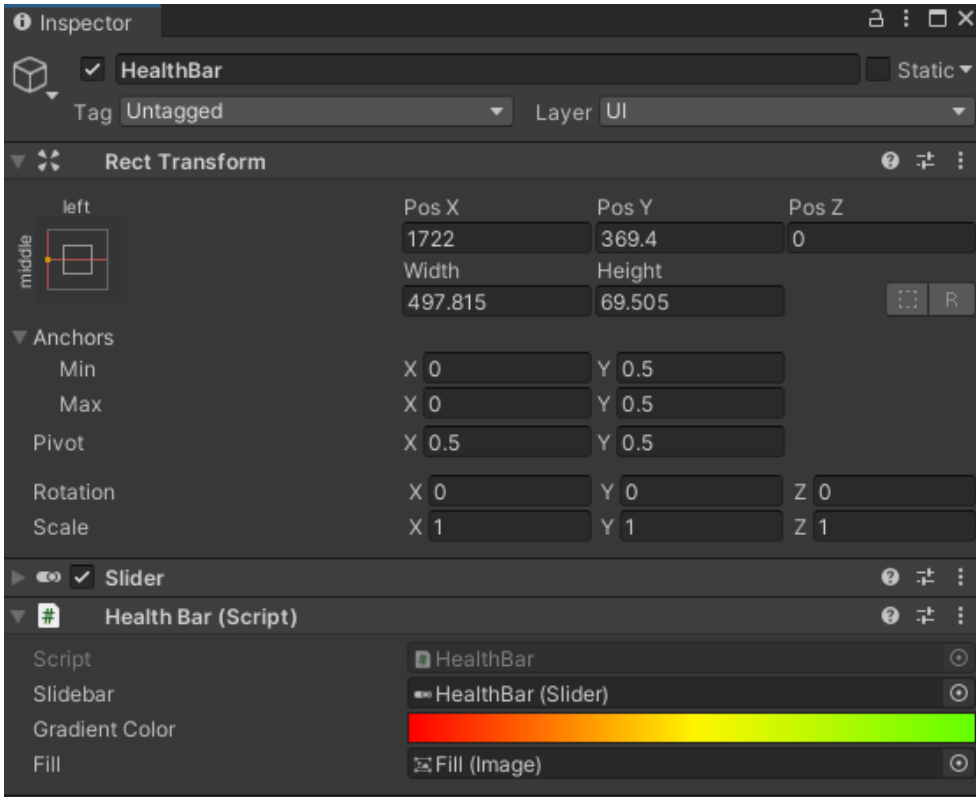


Figure 5.3: Health Bar Properties.

5.3.5 Database

For storing the highest score in the game PlayerPrefs were used. PlayerPrefs is a class that stores player preferences between game sessions. It can store string, float and integer values into the user's platform registry.[21] It was chosen because it's easy and fits our usage.

5.3.2 Scenes

We primarily used the Unity Assets Store to create the scene Interface.

The Assets we have used:

1. **Fantasy landscape:** It is the basic Asset for adding natural details that we used to create roads and landscapes.
2. **Fantasy Skybox FREE:** This Asset is for incorporating and adding a natural sky.
3. **Photogrammetry stump:** This Asset is for adding stump obstacles.

We have five basic scenes and they are: Game, Main Menu, Help, Select Levels and Game Over. All scenes except Game scene Kinect sensor detects the hands only and displays them on the screen. For Game scenes the Kinect sensor detects the joints of the player and reflects them on the character. Also we used three brushes to build the scene.

These brushes are:

1. **Paint Terrain:** This brush is used to build the ground in the scene, in this brush we have imported the green terrain and the brown road from Fantasy landscape Asset.

Figure 5.4 shows the paint terrain attributes.

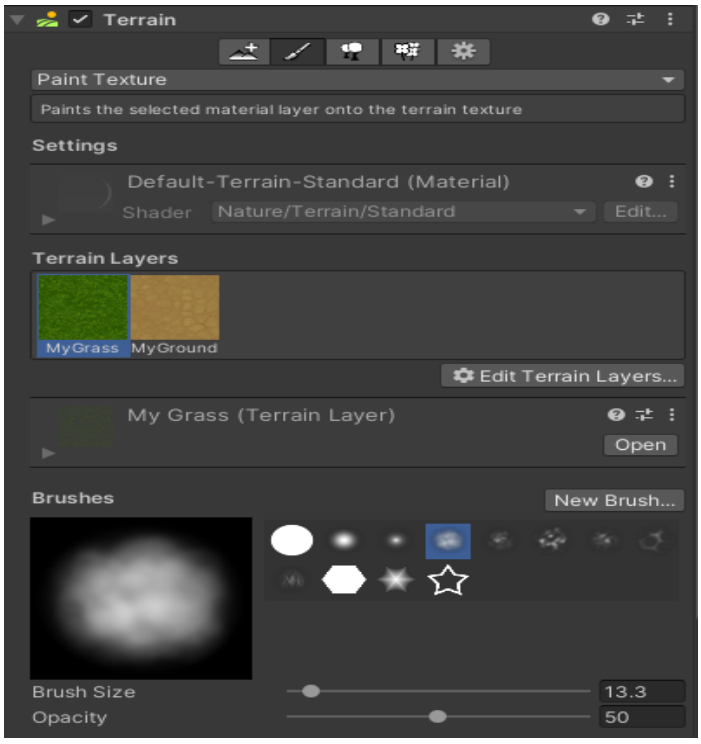


Figure 5.4: Paint Terrain.

2. Paint Trees

This brush is used to build the trees in the scene, so we have imported all trees from Fantasy landscape Asset.

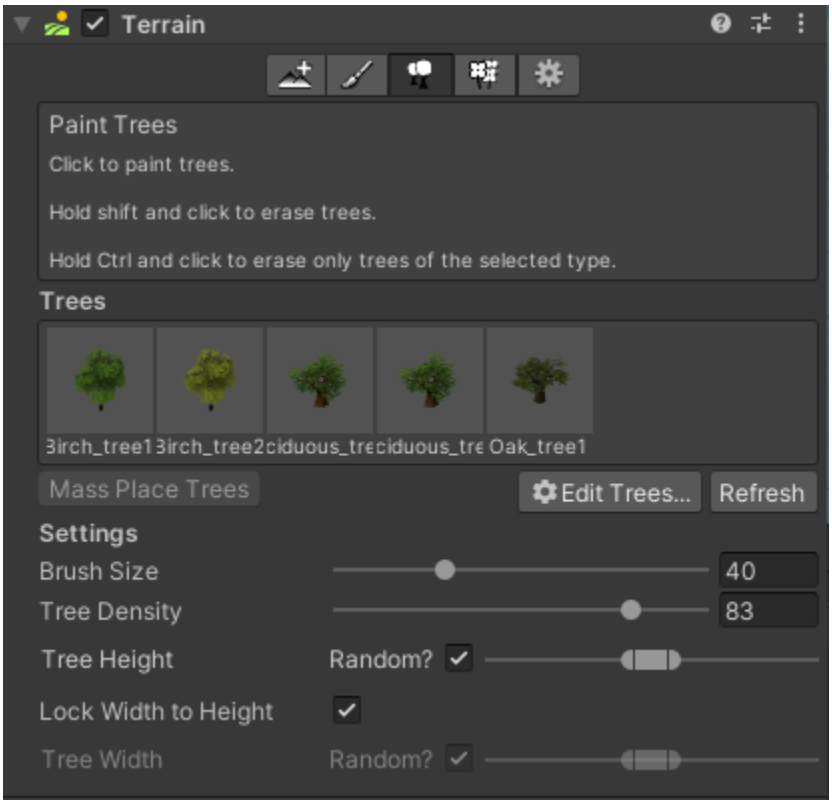


Figure 5.5: Paint Trees.

3. Paint Details

This brush is used to build the details (flowers and rocks) in the scene, so we have imported the flowers, rocks and grass from Fantasy landscape Asset.

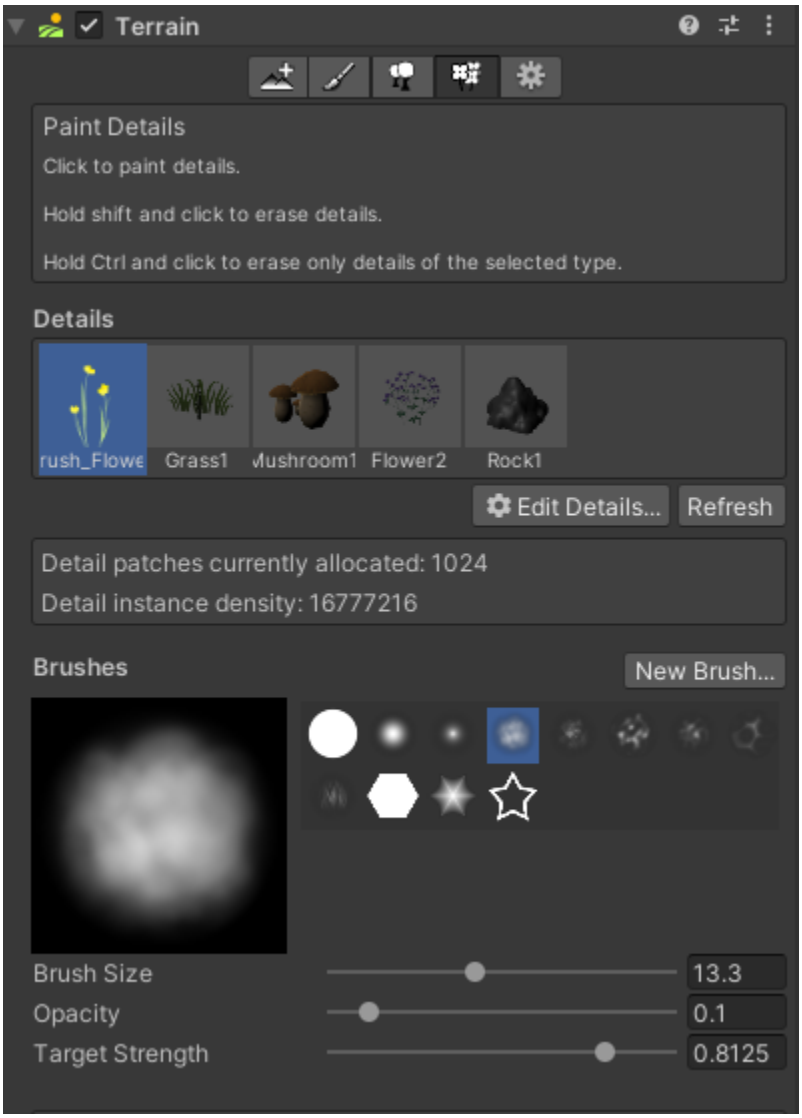


Figure 5.6: Paint Details.

5.4 Implementation Issue

The challenges we faced while working on the project can be restricted to two challenges. The first challenge was the lack of knowledge in the Unity engine. Because it was our first time working with it and it was a little different than web or mobile development. As a result, we used the learning by doing method. The second is the scarcity of information on how to use the Kinect sensor in Unity online.

5.5 Game View

As mentioned we have five basic scenes and here are pictures from each of them:

5.5.1 Game Screen

During the game the player can see his actions reflected on the character. Also he will be able to see his current score and the highest score during the whole games played. Moreover he can see his health at the top of the screen.

Here are two pictures from the game screen:

1. **During Playing**

Figure 5.7: Game Screen.

2. **Player Hits an Obstacle**

Figure 5.8: Hitting Obstacle.

3. **Game Over Screen**

Figure 5.9: Game Over.

4. **Main Menu Screen**

Figure 5.10: Main Menu.

5. **Select Level Screen**

Figure 5.11: Select Level.

6. **Help Screen**

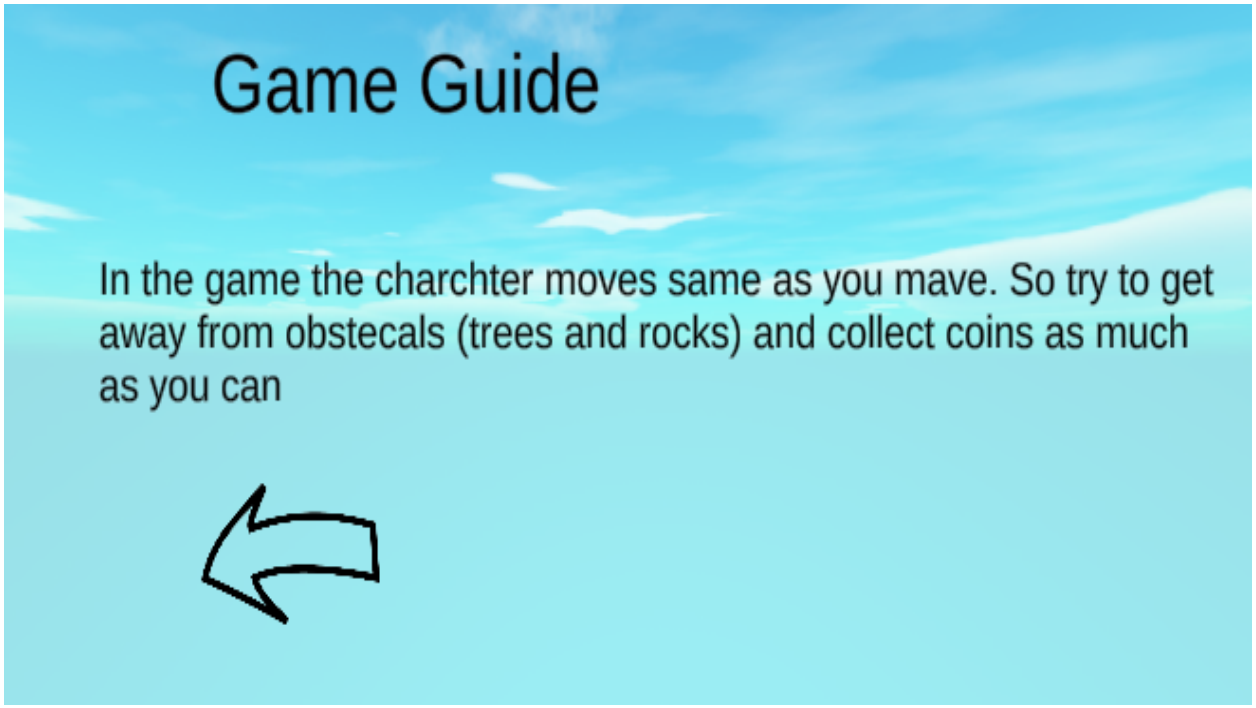


Figure 5.12: Game Guide.

7. **Pause Screen**

Figure 5.13: Pause Screen.

Chapter 6: Testing

6.1 Introduction

This chapter will go over testing for functional requirements and sensor testing.

To ensure that our game works as expected and to assist in managing new changes in specification or implementation.

6.2 Functional Requirements Testing

6.2.1 Scenes Transformations

Table 6.1: Scenes Transformations Test Cases.

Test Case	Scenario	Input data	Expected Output	Actual Output	Test result
Start game button.	The player touches the middle of the start button with the hand displayed on the screen	Player's hand movement	Go to the levels menu screen	Levels menu screen was displayed	Pass
Exit the game.	The player touches the middle of the exit button with the hand displayed on the screen	Player's hand movement	Exit the game	The game is closed	Pass
View help information.	The player touches the middle of the help button with the hand displayed on the screen	Player's hand movement	Go to the help screen	Help screen was displayed	Pass
Back arrow	The player touches the middle of the back arrow with the hand displayed on the screen	Player's hand movement	Go to the main menu screen	Main menu screen was displayed	Pass

Select easy level	The player touches the middle of the easy button with the hand displayed on the screen	Player's hand movement	Go to the game screen and the character will move when the sensor detects player's movement with easy speed (low)	Game screen was displayed and the speed of the character was for easy speed	Pass
Select medium level	The player touches the middle of the medium button with the hand displayed on the screen	Player's hand movement	Go to the game screen and the character will move when the sensor detects player's movement with medium speed (middle)	Game screen was displayed and the speed of the character was for medium speed	Pass
Select hard level	The player touches the middle of the hard button with the hand displayed on the screen	Player's hand movement	Go to the game screen and the character will move when the sensor detects player's movement with hard speed (high)	Game screen was displayed and the speed of the character was for hard speed	Pass

6.2.2 Game Screen

Table 6.2: Game Screen Test Cases.

Test Case	Scenario	Input data	Expected Output	Actual Output	Test result
Earn points.	The characters hits a coin	Player's movement	The score at the top of the screen will increase by one and the coin will disappear. And a coins sound will be played	The score at the top of the screen increased by one and the coin disappeared. And sound was played	Pass
Hit obstacles	The characters hits an obstacle	Player's movement	The health of the player will decrease by 4% and a red screen appears for seconds. And a hit sound will be played	The health of the player decreased by 4% and a red screen appeared for seconds. And sound was played	Pass
View the health.	The player can view his health during the game	-	The player can view a health bar at the top of the game screen	The player viewed a health bar at the top of the game screen	Pass
view highest score.	The player can view his highest score during the game	-	The player can view the highest score at the top of the game screen	The player can viewed the highest score at the top of the game screen	Pass
View current score	The player can view his current score during the game	-	The player can view his score at the top of the game screen	The player can viewed his score at the top of the game screen	Pass

6.2.3 Game Over Screen

Table 6.3 Game Over Screen Test Cases.

Test Case	Scenario	Input data	Expected Output	Actual Output	Test result
Game Over	The character's health is 0	-	Go to the game over screen and display the right score	Game over screen was displayed and the right score was displayed too	Pass
	The character reaches the end of the road	-	Go to the game over screen and display the right score	Game over screen was displayed and the right score was displayed too	Pass
Restart the game button	The player touches the middle of the main menu button with the hand displayed on the screen	Player's hand movement	Go to the game screen with the same selected level before	Game screen was displayed with the right level	Pass
Main menu button	The player touches the middle of the main menu button with the hand displayed on the screen	Player's hand movement	Go to the main menu screen	Main menu screen was displayed	Pass

6.3 Sensor Testing

Table 6.4 Sensor Testing Test Cases.

Test Case	Scenario	Input data	Expected Output	Actual Output	Test result
The sensor is not connected to the device.	The sensor is not connected when the player moves to the game screen	-	The game will not begin and a pause screen will be displayed until the sensor is connected again	The game didn't not begin and a pause screen was displayed until the sensor was connected again	Pass
	The sensor during the game is disconnected	-	The game will pause and a pause screen will be displayed until the sensor is connected again	The game paused and the pause screen was displayed until the sensor was connected again	Pass
Player detection	The player moves away from the sensor	-	The game will stop and a pause screen will be displayed	The game stopped and a pause screen was displayed	Pass
	The sensor detects more than one player	-	The game will stop and a pause screen will be displayed	The game stopped and a pause screen was displayed	Pass
	The sensor detects one player	Player's movement	The game will start normally and the player's movement will be reflected	The game started normally and the player's movement was reflected	Pass

Table 6.5 Sensor Test Cases.

Test Case	Scenario	Input data	Expected Output	Actual Output	Test result
Hand movement	The player moves his hand to choose a button	Player's hand movement	The sensor reflects the player hand movement	The sensor reflected the player hand movement	Pass
	The sensor can't detect both hands of the player	-	Only if there is at least one hand detected the hand in the game will be displayed	Only if there is at least one hand detected the hand in the game will be displayed	Pass
Character movement	The player moves right	Player's movement	The character moves right and the skateboard moves with him	The character moved right and the skateboard moved with him	Pass
	The player moves left	Player's movement	The character moves left and the skateboard moves with him	The character moved left and the skateboard moved with him	Pass
	The player jumps	Player's movement	The character jumps and the skateboard jumps with him	The character jumped and the skateboard jumped with him	Pass
	The player moves his hand	Player's movement	The character moves his shoulders same as the player	The character moved his shoulders same as the player	Pass

6.4 Sounds Testing

Table 6.6 Sounds Test Cases.

Test Case	Scenario	Input data	Expected Output	Actual Output	Test result
Background Music	Moving between scenes and stay in the game for more than 5 minutes	-	Background sound should play in all scenes and repeat when it finish	Background sound played in all scenes and repeated when it finished	Pass

Chapter 7: Conclusion and Future Work

7.1 Conclusion

In the end, we have developed an interactive game using the Kinect Sensor with the goal of developing children's movement and encouraging them to move rather than sitting for long periods of time in front of a phone screen or even TV and sitting incorrectly, which affects their body growth in an unfavorable way, and maintaining a safe distance between them and the screen.

7.2 Future Work

In the near future, we will improve the game by adding some educational features by creating several levels, each of which will be dedicated to learning a specific subject, such as one for learning shapes, colors, and numbers and another for Arabic letters, and so on. Also, we will design the interface to correspond to the current level, for example, the level of shapes; the interface will be linked to the appearance of specific shapes for the player by hearing the name of each shape when it appears. Also we will make it a multiplayer player game such that two players can stand beside each other and play at the same time so this will make it more challenging and fun.

References

[1] Saint, Catherine. “Many Children Under 5 Are Left to Their Mobile Devices, Survey Finds (Published 2015).” *The New York Times*, 2 November 2015, https://www.nytimes.com/2015/11/02/health/many-children-under-5-are-left-to-their-mobile-devices-survey-finds.html. Accessed 24 April 2022.

[2] Richtel, Matt. “Children's Screen Time Has Soared in the Pandemic, Alarming Parents and Researchers (Published 2021).” *The New York Times*, 17 January 2021, https://www.nytimes.com/2021/01/16/health/covid-kids-tech-use.html. Accessed 24 April 2022.

[3] Rothman, Wilson. “Kinect and your kids: What works, what won't.” *NBC News*, 10 November 2010, https://www.nbcnews.com/id/wbna40097124. Accessed 24 April 2022.

[4] Wolf, Mark JP, and Bernard Perron. “The Video Game Theory Reader | Mark JP Wolf, Bernard Perron | Taylo.” *Taylor & Francis eBooks*, 31 May 2013, https://www.taylorfrancis.com/books/edit/10.4324/9780203700457/video-game-theory-reader-mark-wolf-bernard-perron. Accessed 13 May 2022.

[5] “Beat Saber on Steam.” *Steam*, https://store.steampowered.com/app/620980/Beat_Saber/. Accessed 13 May 2022.

[6] “Kinect.” *Wikipedia*, https://en.wikipedia.org/wiki/Kinect. Accessed 24 May 2022.

[7] Ghotkar, Archana. “System block diagram | Download Scientific Diagram.” *ResearchGate*, https://www.researchgate.net/figure/System-block-diagram_fig2_260147325. Accessed 13 May 2022.

[8] Rodriguez, Raul V. “Kinect Sensor: The AI Tool You Did Not Know You Had.” *Analytics India Magazine*, 20 July 2020, https://analyticsindiamag.com/kinect-sensor-the-ai-tool-you-did-not-know-you-had/. Accessed 13 May 2022.

[9] “Setup tips for your Kinect sensor and play space.” *Xbox Support*, https://support.xbox.com/en-US/help/hardware-network/kinect/kinect-sensor-setup-tips. Accessed 24 December 2021.

[10] “The Accuracy of the Microsoft Kinect V2 Sensor.” *ProQuest*, https://www.proquest.com/scholarly-journals/accuracy-microsoft-kinect-v2-sensor-human-gait/docview/2432758986/se-2. Accessed 24 April 2022.

[11] “Why we all need green in our lives.” *CNN*, 5 June 2017, https://edition.cnn.com/2017/06/05/health/colorscope-green-environment-calm/index.html. Accessed 20 April 2022.

[12] “Download Kinect for Windows SDK 2.0 from Official Microsoft Download Center.” *Microsoft*, 21 October 2014, https://www.microsoft.com/en-us/download/details.aspx?id=44561. Accessed 6 May 2022.

[13] “Common game development terms and definitions | Game design vocabulary.” *Unity*, https://unity.com/how-to/beginner/game-development-terms. Accessed 14 April 2022.

[14] “Kinect - Windows app development.” *Microsoft Developer*, https://developer.microsoft.com/en-us/windows/kinect/. Accessed 6 May 2022.

[15] “Common game development terms and definitions | Game design vocabulary.” *Unity*, https://unity.com/how-to/beginner/game-development-terms. Accessed 14 April 2022.

[16] Jordan, Justine. “Is C# a Good Tool for Game Development?” *NarraSoft*, 12 September 2020, https://narrasoft.com/is-c-a-good-tool-for-game-development/. Accessed 14 April 2022.

[17] “Visual Studio Code.” *Wikipedia*, https://en.wikipedia.org/wiki/Visual_Studio_Code. Accessed 14 April 2022.

[18] “What Is GitHub? A Beginner's Introduction to GitHub.” *Kinsta*, 28 May 2021, https://kinsta.com/knowledgebase/what-is-github/. Accessed 20 April 2022.

[19] *Mixamo*, https://www.mixamo.com/#/. Accessed 5 May 2022.

[20] “Manual: Prefabs.” *Unity - Manual*, 7 May 2022, https://docs.unity3d.com/Manual/Prefabs.html. Accessed 11 May 2022.

[21] “Unity - Scripting API: PlayerPrefs.” *Unity - Manual*, https://docs.unity3d.com/ScriptReference/PlayerPrefs.html. Accessed 6 May 2022.